POO en Java

v 0.2

Table of Contents

I - Syntaxe de base

Mathématiques de base et langage Java

Mathématique	Java
$x \times 5$	`x*5`
$a = b$ (test)	`a == b`
$a \neq b$	`a != b`
$a \leq b$	`a <= b`
$a \geq b$	`a >= b`
$a \leq b \leq c$	`a <= b && b <= c`
$a < b < c$	`a < b  && b < c`
$a = b = c$	`a == b && b == c`
VRAI	`true`
FAUX	`false`
ET	`&&`
OU	`||`
$a < b$ **ET** $b < c$	`a < b && b < c`
$a = b$ **ET** $b = c$	`a == b && b == cl`
$a = b$ **OU** $b = c$	`a == b || b == c`
$33 + \frac{5 \times x + 40 \times y}{3}$	`33+(5*x+40*y)/3`
3 est-il divisible par 2 ?	`3 % 2 == 0`

Fonction et appel de fonction

```
public class MemoBase {

	public static float f(float x) {
		return x+1;
	}

	public static void main(String[] args) {
		float x = f(1);
		System.out.println(x);
	}

}
```

Fonction en appelant d'autres

```
public static float g(float x) {
	return x+1;
}

public static float h(float x) {
	return x*x;
}

public static float f(float x) {
	return g(x) + h(x);
}
```

Condition si / sinon

```
public static float f(char c) {
	if (c == 'a') {
		return 1;
	}
	else {
		return 0;
	}
}
```

Condition si / sinon si / sinon

```
public static void e(double x) {
	if (x < 0) {
		System.out.println("negatif.");
	}
	else if (x == 0) {
		System.out.println("zero.");
	}
	else {
		System.out.println("positif.");
	}
}

public static void main(String[] args) {
	e(1);
}
```

Encadrement

```
	public static boolean encadrement(double x, double a, double b)
{
		if (a <= x && x<= b) {
			return true;
		}
		else {
			return false;
		}
	}
```

Utilisation d'un « ou » logique

```
	public static boolean toto(char x, char a, char b) {
		if (x == a || x == b) {
			return true;
		}
		else {
			return false;
		}
	}

	public static void main(String[] args) {
		System.out.println(toto('y', 'c', 'y'));

	}
/*
affichage : true
*/
```

Différence entre fonction et procédure

```
	public static int toto(int x, int y) {
		return x+y;
	}

	public static void main(String[] args) {
		System.out.println(toto(1, 2));
	}
```

```
	public static void toto(int x, int y) {
		System.out.println(x+y);
	}

	public static void main(String[] args) {
		toto(1, 2);
	}
```

Pseudo-code et Java en vis-à-vis

```
fonction toto(x)
    si x=="toto" alors
        retourner VRAI
    sinon:
        retourner FAUX
    fin si
```

```
public static boolean toto(String s)
{
        if (s.equals("toto"))
                return true;
        else
                return false;
}
```

Boucle « pour » : afficher les entiers de 0 à n-1

```
procédure afficher(n)
    pour i = 0, ..., n-1
alors
        affichage(i)
    fin pour
```

```
public static void afficher(int n) {
        for (int i=0; i<n; i++) {
                System.out.println(i);
        }
}
```

Boucle « tant que » : afficher les entiers de 0 à n-1

```
procédure afficher(n)
    i = 0
    tant que i <= n-1 faire
        affichage(i)
        i = i+1
    fin tant que
```

```
public static void afficher(int n) {
        int i=0;
        while (i<n) {
                System.out.println(i);
                i++;
        }
}
```

Affichage des éléments d'un tableau

```
public static void afficher(int tab[]) {
      int n = tab.length;
      for (int i=0; i<n; i++)
            System.out.print(tab[i] + " ");
      System.out.println();
}

public static void main(String[] args) {
    int t[] = new int[4];
    t[0] = 4;
    t[1] = 3;
    t[2] = 2;
    t[3] = 1;
    afficher(t);
}
```

Somme des éléments d'un tableau

```
public static double somme(double tab[]) {
      int n = tab.length;
      double res = 0;
      for (int i=0; i<n; i++)
```

```
            res = res + tab[i];
      return res;
}

public static void main(String[] args) {
      double t[] = new double[4];
    t[0] = 4.3;
    t[1] = 3.4;
    t[2] = 2.5;
    t[3] = 1.3;
    System.out.println(somme(t));
}
```

Affichage simultané de deux tableaux

```
public static void afficher2(int tab1[], int tab2[]) {
      if (tab1.length != tab2.length)
            return ;
      int n = tab1.length;
      for (int i=0; i<n; i++)
            System.out.println(tab1[i]+ ", "+ tab2[i]);
}
```

Somme de deux tableaux terme à terme dans un troisième

```
public static int [] somme2(int tab1[], int tab2[]) {
      // on suppose tab1.length = tab2.length
      int n = tab1.length;
      int tab3[]  = new int[n];
      for (int i=0; i<n; i++)
            tab3[i] =tab1[i]+tab2[i];
      return tab3;
}
```

Afficher le k-ième élément d'un tableau

```
static char kIeme(char tab[], int j) {
      int n = tab.length;
      // index bien défini ?
      if (0<=j && j<=n-1)
                  return tab[j];
      return 0; /* on utilise ici le 0 pour indiquer
                   un probleme */
}
```

Maximum des éléments d'un tableau

```
static double maxi(double tab[]) {
	int n = tab.length;
	if (n == 0) return -1; // cas du tableau vide
	double res = tab[0];
	for (int i=1; i<n; i++) {
		if (tab[i] > res)
			res = tab[i];
	}
	return res;
}
```

Indice d'un élément maximum d'un tableau

```
static double iMaxi(double tab[]) {
	int n = tab.length;
	if (n == 0) return -1; // cas du tableau vide
	double res = tab[0];
	int iRes = 0;
	for (int i=1; i<n; i++) {
		if (tab[i] > res) {
			res = tab[i];
			iRes = i;
		}
	}
	return iRes;
}
```

Afficher les caractères d'une chaîne de caractères

```
static void afficherChar(String s) {
	for (int i=0; i<s.length(); i++)
		System.out.println(s.charAt(i));
}
```

Symbole mathématique : somme ∑

$\sum_{i=0}^{n-1} i = 0+1+2+3+...+(n-1)$	`static int somme(int n) {`
	`  int res = 0;`
	`  for (int i=0; i<n; i++)`
	`  res += i;`
	`  return res;`
	`}`

$\sum_{i=0}^{n-1} \cos(i) = \cos(0)+\cos(1)+\cos(2)+...+\cos(n-1)$	`static double sommeCos(int n) {`
	`  double res = 0;`
	`  for (int i=0; i<n; i++)`
	`  res += Math.cos(i);`
	`  return res;`
	`}`

Symbole mathématique : produit ∏

$\prod_{i=1}^{n-1} i = 1 \times 2 \times 3 \times ... \times (n-1)$	`static int produit(int n) {`
	`  int i;`
	`  int res=1;`
	`  for (i=1; i<n; i++)`
	`  res *= i;`
	`  return res;`
	`}`

$\prod_{i=0}^{n-1} e^i = e^0 \times e^1 \times e^2 \times ... \times e^{n-1}$	`static double produitExp(int n)`
	`{`
	`  double res = 1;`
	`  for (int i=0; i<n; i++)`
	`  res *= Math.exp(i);`
	`  return res;`
	`}`

II - Conventions de nommage

Type	Exemples
Classe	`class Personne;` `class GroupePersonnes;`
Méthode	`Deplacer();` `afficherMaximum();`

Variable	`int i;`
	`char c;`
	`float scoreMax;`

Constante	`static final int LARGEUR_MAX = 999;`

III - Visibilité en Java

Identificateur de visibilité	Signification
`private (-)`	Accessible seulement à la classe.
`package friendly ()`	Valeur par défaut, accessible aux classes dans le même paquetage.
`protected (#)`	Accessible aux classes dans le même paquetage et à toute classe dérivée en dehors du paquetage.
`public (+)`	Accessible à toutes les classes.

IV - Getters imbriqués

```
Class Ville {
   - maire : Maire;
   + getMaire() : Maire;
}


Class Maire {
   - nom : String;
   - parti : PartiPolitique;
   + getNom() : String;
   + getParti() : PartiPolitique;
}


Class PartiPolitique {
   - String nom;
   + getNom() : String;
}
```

Soit ***city*** de type ***Ville***.

Obtenir le maire de la ville ***city*** :

→ `city.getMaire()`

Obtenir le nom du maire de la ville ***city*** :

→ `city.getMaire().getNom()`

Obtenir le nom du parti politique du maire de la ville ***city*** :

→ `city.getMaire().getParti().getNom()`